**Spark Dashboard Project Overview**

**Introduction**

The overall focus of this project is to create a real-time analytics dashboard of access and break-in attempts to webservers. This document summarizes the current state of the project as of the end of the Fall ’21 semester.

**Architecture**

**Webserver**

Node: ws1

**Grafana**

Node: Grafana

Read data from InfluxDB and display on Grafana Dashboard

Write data to InfluxDB on Grafana Node

**Master**

Node: head

Auth log data is parsed and analyzed using Spark cluster (not yet fully implemented)

Spark Workers

Node: Worker-3

Node: Worker-2

Node: Worker-1

Authentication log messages are read from broker

**Kafka Broker**

Node: broker

Webserver write authentication logs to Kafka Broker in real time

At a high level, the architecture consists of a honeypot webserver(s), a data parsing/processing pipeline, and a data visualization dashboard. The current setup comprises one webserver, a [Spark](https://spark.apache.org/) cluster for data analysis, a [Kafka](https://kafka.apache.org/) broker to facilitate data streaming from the webserver to the Spark cluster, a Grafana data visualization dashboard, and an [InfluxDB](https://grafana.com/docs/grafana/latest/datasources/influxdb/) database to stage data for the [Grafana](https://grafana.com/) dashboard.

The system is hosted on a Cloudlab cluster consisting of seven Linux compute nodes:

* **ws1**: hosts the honeypot webserver
* **head**: serves as head node of the Spark cluster and the main compute node for data parsing and analysis
* **broker**: hosts a Kafka broker for transfer of data from webserver to head
* **3 worker nodes**: serve as nodes in the Spark cluster – execute computational tasks for Spark jobs
* **grafana**: hosts the InfluxDB database and the Grafana dashboard

**Data Flow**

The webserver has a record of authentication events in /var/log/auth.log. [log\_writer.py](https://github.com/WCU-EDGE/spark-dashboard/blob/main/code/log_writer.py) writes this authentication log to a Kafka topic called “auth.log”, hosted on the broker node. log\_writer.py sends new messages from the auth.log file to the Kafka topic as they come in.

A script running on the head node reads log messages from the Kafka topic using the Kafka Consumer API (see [consumer.ipynb](https://github.com/WCU-EDGE/spark-dashboard/blob/main/notebooks/consumer.ipynb)). The script then parses and analyzes the log data. This is still very much a work in progress – there is no holistic script implemented for this. consumer.ipynb is a proof-of-concept for reading log messages via Kafka Consumer, and the first two cells of [spark-streaming-test.ipynb](https://github.com/WCU-EDGE/spark-dashboard/blob/main/notebooks/spark-streaming-test.ipynb) setup a Spark context for data processing using the Cloudlab cluster, although the configuration still needs to be adjusted to utilize the worker nodes. After the data is processed on the head node, it is written to an influxDB database on the Grafana node (also showcased in commented lines of consumer.ipynb) . A final version of this script/notebook would read log messages from the Kafka topic (perhaps with [spark streaming](https://spark.apache.org/docs/latest/streaming-programming-guide.html)), parse the data and create analytics using PySpark, and then write the output data to influxDB.

The Grafana Dashboard then reads the data from influxDB and displays it.

**Setup**

Most of our progress during the Fall ’21 semester was in the infrastructure/deployment of the Cloudlab cluster. When the cluster is activated, each node needs to have its processes initialized and their respective dependencies installed, as these do not persist between Cloudlab sessions. Each node has a *setup\_\_\_\_\_.sh* script to handle this. The Cloudlab setup and execution of the setup scripts is orchestrated by *profile.py*.

**Other Components**

-Grafana server webUI

Spark Cluster webUI

Jupyter Notebook server

**Current/Unresolved Issues**

**Scrapped Ideas**

-Prometheus

-Writing back to Broker before InfluxDB